**Topic 2 DQ 1**

What are the key steps involved in creating a secure login and registration system in an ASP.NET Core MVC application? Discuss the importance of hashing and salting passwords and how they enhance security.

Creating a secure login and registration system in an ASP.NET Core MVC application involves several key steps that focus on protecting user credentials and ensuring data integrity. Here are the essential steps:

1. **User Registration**:
   * Collect user information, such as email and password.
   * Generate a unique salt for each user to enhance security.
   * Hash the password combined with the salt using a secure hashing algorithm (e.g., PBKDF2, BCrypt, or Argon2) before storing it in the database. This ensures that even if the database is compromised, the original passwords cannot be easily retrieved [[1]](https://code-maze.com/csharp-hashing-salting-passwords-best-practices/)[[3]](https://learn.microsoft.com/en-us/aspnet/core/security/data-protection/consumer-apis/password-hashing?view=aspnetcore-9.0).
2. **Password Hashing and Salting**:
   * Hashing transforms the plain-text password into a fixed-length string that cannot be reversed. This process is crucial because it protects passwords from being exposed in their original form.
   * Salting involves adding a unique, random string (the salt) to the password before hashing. This ensures that even if two users have the same password, their hashes will differ due to the unique salts. This practice mitigates risks associated with precomputed hash attacks, such as rainbow tables [[1]](https://code-maze.com/csharp-hashing-salting-passwords-best-practices/)[[2]](https://www.c-sharpcorner.com/blogs/creating-a-secure-password-storage-system-in-asp-net-web-forms-with-sql2).
3. **User Login**:
   * When a user attempts to log in, retrieve the stored hashed password and salt from the database.
   * Hash the entered password using the same salt and compare the result with the stored hash. If they match, the login is successful; otherwise, it fails. This method ensures that the original password is never stored or transmitted in plain text [[2]](https://www.c-sharpcorner.com/blogs/creating-a-secure-password-storage-system-in-asp-net-web-forms-with-sql2)[[3]](https://learn.microsoft.com/en-us/aspnet/core/security/data-protection/consumer-apis/password-hashing?view=aspnetcore-9.0).
4. **Implementing Security Best Practices**:
   * Use HTTPS to encrypt data in transit, protecting user credentials from interception.
   * Implement account lockout mechanisms after a certain number of failed login attempts to prevent brute-force attacks.
   * Regularly update and patch the application to protect against known vulnerabilities [[1]](https://code-maze.com/csharp-hashing-salting-passwords-best-practices/)[[2]](https://www.c-sharpcorner.com/blogs/creating-a-secure-password-storage-system-in-asp-net-web-forms-with-sql2).

The importance of hashing and salting passwords cannot be overstated. Hashing ensures that even if an attacker gains access to the database, they cannot easily retrieve the original passwords. Salting adds an additional layer of security by ensuring that identical passwords do not produce the same hash, making it significantly harder for attackers to use precomputed hash tables to crack passwords. Together, these techniques enhance the overall security of user credentials and help maintain user trust in the application [[1]](https://code-maze.com/csharp-hashing-salting-passwords-best-practices/)[[3]](https://learn.microsoft.com/en-us/aspnet/core/security/data-protection/consumer-apis/password-hashing?view=aspnetcore-9.0).

1. [Hashing and Salting Passwords in C# - Best Practices - Code Maze](https://code-maze.com/csharp-hashing-salting-passwords-best-practices/)
2. [Creating a Secure Password Storage System in ASP.NET Web Forms with SQL](https://www.c-sharpcorner.com/blogs/creating-a-secure-password-storage-system-in-asp-net-web-forms-with-sql2)
3. [Hash passwords in ASP.NET Core | Microsoft Learn](https://learn.microsoft.com/en-us/aspnet/core/security/data-protection/consumer-apis/password-hashing?view=aspnetcore-9.0)